

# An introduction to SDE simulation

Simon J.A. Malham · Anke Wiese

5th April 2010

**Abstract** We outline the basic ideas and techniques underpinning the simulation of stochastic differential equations. In particular we focus on strong simulation and its context. We also provide illustrative examples and sample matlab algorithms for the reader to use and follow. Our target audience is advanced undergraduate and graduate students interested in learning about simulating stochastic differential equations. We try to address the FAQs we have encountered.

**Keywords** stochastic simulation

**Mathematics Subject Classification (2000)** 60H10 · 60H35

## 1 Introduction

### 1.1 When is a model stochastic?

Often in modelling, we need to incorporate a phenomenon or influence that seems random, whose behaviour we can only recognize as statistically Gaussian. The prototypical example is behaviour of molecular origin—the Brownian motion of a pollen particle on a water surface. However the influences can be large scale, for example a turbulent wind or atmospheric flow, or thousands of people buying and selling millions of shares.

Imagine tracking and adjusting the flight of a rocket after lift-off. If the rocket is buffeted by a random turbulent wind, you might sensibly equip the rocket with stabilizers that kick-in if a gust diverts it too far. Computing the path of the rocket, and regulating it to ensure it threads pre-arranged target positions at critical junctures (eg. stage separation), is a stochastic simulation problem. Indeed it is a *strong simulation*

---

Simon J.A. Malham · Anke Wiese  
 Maxwell Institute for Mathematical Sciences  
 and School of Mathematical and Computer Sciences  
 Heriot-Watt University, Edinburgh EH14 4AS, UK  
 Tel.: +44-131-4513200  
 Fax: +44-131-4513249  
 E-mail: S.J.Malham@ma.hw.ac.uk  
 E-mail: A.Wiese@ma.hw.ac.uk

*problem*, as conditional on the path, the stabilizers will affect the outcome. The pricing of financial derivatives (futures/options) is another example. One tracks a security price (eg. a share price) that is randomly buffeted by market forces. Pricing the derivative you wish to sell, which might be exercised by the buyer at a future time, involves hedging/regulating the proportion of your investment in the security (the rest invested in risk-free bonds) so as to minimize your risk. Building in stabilizers/barriers that kick-in if the security price skews too wildly, is again, a strong stochastic simulation problem.

## 1.2 What is a stochastic differential equation?

Consider a model that incorporates some random phenomena whose statistics is Gaussian. Suppose the state of the system is recorded through the vector  $y_t \in \mathbb{R}^N$ , with  $N \geq 1$ . Suppose there are several random sources, say  $W^1, \dots, W^d$ ; these are Wiener processes; think of them as independent, continuous, nowhere differentiable, functions of time. Indeed the time derivatives of the Wiener processes represent pure white noise. Suppose the affect of the Wiener processes on the model, i.e. which way they skew the solution, is recorded through the vector fields  $V_1, \dots, V_d$ . Without the noise we would have a nice fuzz-free signal which is generated by the vector field  $V_0$ . A stochastic model for the system state  $y_t \in \mathbb{R}^N$  might be that it evolves according to the stochastic differential equation:

$$\frac{dy}{dt} = V_0(y_t) + V_1(y_t) \frac{dW_t^1}{dt} + \dots + V_d(y_t) \frac{dW_t^d}{dt}.$$

This representation of the model is somewhat formal; after all the pure white noise terms  $dW_t^i/dt$  need to be interpreted in an extremely weak sense, we prefer to represent the model in the form

$$dy_t = V_0(y_t) dt + V_1(y_t) dW_t^1 + \dots + V_d(y_t) dW_t^d.$$

Indeed there is also a representation in a preferred integral form which we meet presently. With this in mind though, an important observation at this point, is to recall that Wiener processes are continuous functions. Thus the solution function will also be continuous.

**Example (Langevin equation/Brownian motion).** Consider the equation of motion of a pollen particle suspended in a fluid flow. The particle might obey the following equation of motion for its velocity  $y_t$ :

$$\frac{dy_t}{dt} = -a y_t + \sqrt{b} \frac{dW_t}{dt},$$

where  $a$  and  $b$  are constants. The right-hand side is the force exerted on the particle per unit mass. There is a deterministic force  $-a y_t$  and a white noise force  $\sqrt{b} dW_t/dt$  supposed to represent the random buffeting by the water molecules.

### 1.3 What information do we want to retrieve?

If the time interval of interest is  $[0, T]$ , and our initial deterministic state is  $y_0$ , each realization  $\omega$  of an individual Wiener path  $W(\omega)$  will produce a different outcome  $y_t(\omega)$  for  $t \in [0, T]$ . Practical information of interest is often expected values of functions  $f$  of the solution,  $f(y_t)$ , or more generally path-dependent functions of the solution  $f(t, y_s; s \leq t)$ . Hence we might want to compute

$$\mathbf{E} f(y_t) := \int f(y_t(\omega)) \, d\mathbf{P}(\omega),$$

where  $\mathbf{P}$  is a probability measure. For example we could pick  $f$  to be of polynomial or exponential form, synonymous with statistical moments of  $y_t$ . If  $f$  is the identity map, we obtain the expectation of the solution. If we take  $f$  to be  $\|\cdot\|_p^p$ , where  $\|\cdot\|_p$  is the  $p$ -vector norm, then we define the  $L^p$ -norm for  $p \geq 1$  by

$$\|y_t\|_{L^p}^p := \int \|y_t(\omega)\|_p^p \, d\mathbf{P}(\omega).$$

### 1.4 How do we retrieve it?

There are two main simulation approaches to extract such information, we can either:

- Solve a *partial differential equation*; or
- Perform *Monte–Carlo simulation*.

Associated with every stochastic differential equation, there is a parabolic *partial differential equation* for  $u(t, y)$  whose solution at time  $t \in [0, T]$  is

$$u(t, y) = \mathbf{E} f(y_t)$$

provided  $u(0, y) = f(y)$  initially. Thus solving the associated partial differential equation on  $[0, T]$  will generate lots of information about the solution to the stochastic differential equation at time  $t$ . By judiciously choosing  $f$  to be a monomial function we can generate any individual moment of the solution  $y_t$  we like, or if we choose  $f = \exp$  we generate all the moments simultaneously (this is essentially the Laplace transform). If we choose  $f$  to be a Dirac delta function we generate the transition probability distribution for  $y_t$ —the probability density function for  $y_t$  conditioned on the initial data  $y_0$ . Choosing  $f$  to be a Heaviside function generates the corresponding (cumulative) distribution function. Of course often, the partial differential equation will have to be solved approximately. Also note, if we fix a form for  $f$  from the start, for example  $f$  as the identity map, then we simply solve an ordinary differential equation for  $u(t, y)$ .

In *Monte–Carlo simulation*, we generate a set of suitable multidimensional sample paths  $\hat{W}(\omega) := (\hat{W}^1(\omega), \dots, \hat{W}^d(\omega))$  on  $[0, T]$ ; in practice,  $\omega$  belongs to a large but finite set. For each sample path  $\hat{W}(\omega)$ , we generate a sample path solution  $\hat{y}(\omega)$  to the stochastic differential equation on  $[0, T]$ . This is often achieved using a truncation of the ‘stochastic’ Taylor series expansion for the solution  $y$  of the stochastic differential equation, on successive small subintervals of  $[0, T]$ . Suppose for example, we wanted to compute the expectation  $\mathbf{E} f(\hat{y}_t)$ . Having generated a set of approximate solutions  $\hat{y}_t(\omega_i)$  at time  $t \in [0, T]$ , for  $i = 1, \dots, P$  with  $P$  large, we can estimate  $\mathbf{E} f(\hat{y}_t)$  by

computing the mean-sum over the large finite set of approximate sample solutions  $\hat{y}_t(\omega_i)$ . Hence in practice we approximate

$$\int f(y_t(\omega)) \, dP(\omega) \approx \frac{1}{P} \sum_{i=1}^P f(y_t(\omega_i))$$

where  $P$  is the total number of sample paths. A natural dichotomy now arises. To compute  $E f(\hat{y}_t)$ , we can in fact choose any suitable multidimensional paths  $\hat{W}(\omega)$  that leave  $E f(\hat{y}_t)$  approximately invariant, in the sense that  $\|E f(y_t) - E f(\hat{y}_t)\|$  is sufficiently small. This is a *weak approximation*. For example, increments  $\Delta W^i$  in each computation interval can be chosen from a suitable binomial branching process, or using Lyons and Victoir's cubature method [28]. Note that since the approximate paths are not close to Brownian paths we cannot compare  $\hat{y}_t$  and  $y_t$  directly. In a *strong approximation*, discrete increments  $\Delta W^i$  in each computation interval are directly sampled from the Gaussian distribution. This is more expensive. However, the sample paths  $\hat{W}(\omega)$  generated in this way, allow us to compare  $\hat{y}_t(\omega)$  and  $y_t(\omega)$  directly in the sense that we can guarantee  $E \|y_t - \hat{y}_t\|$  will be sufficiently small. Naturally, using strong simulation we can also account for path-dependent features, such as conditional cut-offs or barriers, when we investigate individual solutions or the final expectation or higher moments of the approximate solution paths  $\hat{y}$ . For a comprehensive overview of Monte-Carlo methods see Boyle, Broadie and Glasserman [6].

### 1.5 What is required?

In general to extract qualitative and quantitative information from a stochastic differential system requires the languages and techniques of several mathematical disciplines, notably:

1. *Integration*: in Brownian motion new information is continuously generated on infinitesimally small time scales (imagine the pollen particle jiggles); solution as with ordinary differential equations is by integration, except that now the coefficients of the evolution equation—the Wiener processes—are no longer differentiable.
2. *Statistics*: we typically extract statistical information from the solution process;
3. *Geometry*: as with ordinary differential equations, preserving invariant geometric structure of the solution path evolution is important; for example the solution may evolve on a homogeneous manifold;
4. *Simulation*: stochastic differential equations more often than not, are *not* integrable in the classical sense, and require numerical computation.

For general background reading, we recommend as follows. For a comprehensive introduction to the theory underlying stochastic differential equations download Evans' notes [12]. For an introduction to numerical simulation, see Higham's notes [19]. The answer to just about any other question that a beginner may have on numerical simulation, not covered above or here, can likely be found in the treatise by Kloeden and Platen [22].

## 2 Stochastic differential equations

### 2.1 Integral representation

Consider the nonlinear stochastic differential equation of order  $N \in \mathbb{N}$  given by

$$y_t = y_0 + \int_0^t \tilde{V}_0(y_\tau) d\tau + \sum_{i=1}^d \int_0^t V_i(y_\tau) dW_\tau^i.$$

Here  $(W^1, \dots, W^d)$  is a  $d$ -dimensional Wiener process, i.e. there are  $d$  independent driving noisy signals. We assume there exists a unique solution  $y: [0, T] \mapsto \mathbb{R}^N$  for some time interval  $[0, T] \subseteq \mathbb{R}_+$ . We suppose that  $\tilde{V}_0$  and  $V_i: \mathbb{R}^N \rightarrow \mathbb{R}^N$ ,  $i = 1, \dots, d$ , are smooth non-commuting autonomous vector fields. We are representing the stochastic differential equation above in Itô form and indicate this by using  $\tilde{V}_0$  to represent the *Itô drift vector field*. We call the vector fields  $V_i$  for  $i = 1, \dots, d$  associated with the driving noise terms the *diffusion vector fields*. Presently we will distinguish, and explain, the Itô representation as opposed to the Stratonovich representation for a stochastic differential equation. We also remark that a common convention is to set  $W_t^0 \equiv t$ . Results on existence and uniqueness of solutions can be found in Kloeden and Platen [22].

### 2.2 Driving Wiener process

A scalar driving noisy signal or disturbing Brownian motion has a concise definition and set of properties formulated by Wiener.

**Definition 1 (Wiener process)** A scalar *standard Wiener process* or *standard Brownian motion*  $W$  is a continuous process that satisfies the three conditions:

1.  $W_0 = 0$  with probability one;
2.  $W_t - W_s \sim \sqrt{t-s} \cdot \mathbf{N}(0, 1)$  for  $0 \leq s < t$ , where  $\mathbf{N}(0, 1)$  denotes a standard Normal random variable;
3. Increments  $W_t - W_s$  and  $W_\xi - W_\eta$  on distinct time intervals are independent, i.e. for  $0 \leq s < t < \eta < \xi$ .

Note that with probability one an individual Brownian path is nowhere differentiable.

**Example (Langevin equation).** As we have seen, the Brownian motion of a pollen particle suspended in a fluid flow obeys the following equation of motion for its velocity  $y_t$ :

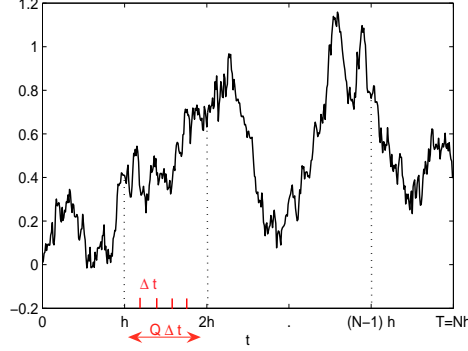
$$dy_t = -a y_t dt + \sqrt{b} dW_t,$$

where  $a$  and  $b$  are constants, and  $W$  is a scalar Wiener process. This type of stochastic differential equation is said to have *additive noise* as the diffusion vector field is constant. It is also an example of an *Ornstein–Uhlenbeck process*.

**Example (scalar linear equation).** Consider the scalar linear stochastic differential equation

$$dy_t = a y_t dt + b y_t dW_t$$

driven by a scalar Wiener process  $W$ , with  $a$  and  $b$  constants. This stochastic differential equation is said to have *multiplicative noise* as the diffusion vector field depends



**Fig. 1** Example scalar Wiener path on the interval  $[0, T]$ .

multiplicatively on the solution  $y_t$ . We can in fact analytically solve this equation, the solution is

$$y_t = y_0 \exp\left(a t + b W_t - \frac{1}{2} b^2 t\right).$$

The additional term  $-\frac{1}{2} b^2 t$  is due to the Itô correction, which we discuss shortly.

### 2.3 Vector fields and flow maps

Consider an ordinary differential equation governed by an autonomous vector field  $V$  that evolves on a homogeneous manifold  $\mathcal{M}$  so that

$$\frac{dy_t}{dt} = V(y_t),$$

with initial data  $y_0 \in \mathcal{M}$ . Here we suppose  $\mathcal{M}$  to be an  $N$ -dimensional manifold. The reader can for simplicity assume  $\mathcal{M} \equiv \mathbb{R}^N$  for the rest of this section if they choose. Let  $\text{Diff}(\mathcal{M})$  denote the group of diffeomorphisms of  $\mathcal{M}$ . The *flow-map*  $\varphi_{t,t_0} \in \text{Diff}(\mathcal{M})$  for the ordinary differential equation above is the map taking the solution configuration on  $\mathcal{M}$  at time  $t_0$  to that at time  $t$ , i.e. it is the map  $\varphi_{t,t_0}: \mathcal{M} \rightarrow \mathcal{M}$  such that

$$\varphi_{t,t_0}: y_{t_0} \mapsto y_t.$$

In other words for any data  $y_0 \in \mathcal{M}$  at time  $t_0$  we can determine its value  $y_t \in \mathcal{M}$  at time  $t$  later by applying the *action* of the flow map  $\varphi_{t,t_0}$  to  $y_0$  so that  $y_t = \varphi_{t,t_0} \circ y_0$ . Note that the flow map satisfies the usual group properties

$$\varphi_{t,s} \circ \varphi_{s,t_0} = \varphi_{t,t_0},$$

with  $\varphi_{t_0,t_0} = \text{id}$ , the identity diffeomorphism. If  $f \in \text{Diff}(\mathcal{M})$ , the chain rule reveals that for all  $y \in \mathcal{M}$ , we have

$$\frac{df(y)}{dt} = V(y) \cdot \partial_y f(y),$$

where  $\partial_y \equiv \nabla_y$  is the usual gradient operator with respect to each component of  $y$ . In other words, vector fields act on the group of diffeomorphisms  $\text{Diff}(\mathcal{M})$  as first order partial differential operators, and for any  $f \in \text{Diff}(\mathcal{M})$ , we write

$$V \circ f \circ y = V(y) \cdot \partial_y f(y).$$

We now think of  $V(y) \cdot \partial_y$  as a first order partial differential operator and an element of the tangent space  $\mathfrak{X}(\mathcal{M})$  to  $\text{Diff}(\mathcal{M})$ . In particular, choose the diffeomorphic map  $f$  to be the flow map  $\varphi_t \equiv \varphi_{t,0}$ . Then for all  $y_0 \in \mathcal{M}$  and  $y_t = \varphi_t \circ y_0$  we have,

$$\frac{d}{dt}(\varphi_t \circ y_0) = V \circ \varphi_t \circ y_0.$$

We pull back this ordinary differential equation for  $y_t \in \mathcal{M}$  to the *linear* functional differential equation in  $\text{Diff}(\mathcal{M})$ :

$$\frac{d\varphi_t}{dt} = V \circ \varphi_t.$$

Since  $\varphi_0 = \text{id}$ , the solution is  $\varphi_t = \exp(tV)$ , giving the representation of the flow-map as the *exponentiation of the vector field*. Hence we see that

$$y_t = \exp(tV) \circ y_0.$$

An important and illustrative concomitant derivation of this result is as follows. Integrating the functional differential equation for the flow-map we get

$$\varphi_t = \text{id} + \int_0^t V \circ \varphi_\tau d\tau.$$

To solve this integral equation, we set up the formal iterative procedure given by

$$\varphi_t^{(n+1)} = \text{id} + \int_0^t V \circ \varphi_\tau^{(n)} d\tau,$$

with  $\varphi_t^{(0)} = \text{id}$ . For example, after two iterations:  $\varphi_t^{(2)} = \text{id} + tV \circ \text{id} + \frac{1}{2}t^2 V^2 \circ \text{id}$ , where  $V^2 \equiv V \circ V$ . Hence in the limit we obtain the exponential form for  $\varphi_t$  above.

The composition of two vector fields  $U$  and  $V$  is a second order differential operator:

$$\begin{aligned} U \circ V &= (U(y) \cdot \partial_y)(V(y) \cdot \partial_y) \\ &= \left( (U(y) \cdot \partial_y)(V(y)) \right) \cdot \partial_y + U(y) \otimes V(y) : \partial_{yy} \\ &= \sum_{i,j=1}^N U^i \partial_{y_i} (V^j) \partial_{y_j} + \sum_{i,j=1}^N U^i V^j \partial_{y_i y_j}. \end{aligned}$$

Importantly we now observe that since  $\partial_{y_i y_j} = \partial_{y_j y_i}$  as operators on  $\text{Diff}(\mathcal{M})$ , the Lie bracket  $[U, V]$  of two vector fields is a vector field:

$$[U, V] = U \circ V - V \circ U = \left( (U(y) \cdot \partial_y)V(y) - (V(y) \cdot \partial_y)U(y) \right) \cdot \partial_y.$$

Note the sum of two vector fields is itself a vector field. Hence the set of vector fields is a *Lie algebra*  $\mathfrak{X}(\mathcal{M})$ —closed under summation and Lie product  $[\cdot, \cdot]$ .

## 2.4 Stratonovich representation

There are two generic representations for stochastic differential equations. One can either express them in Itô form, as we did at the beginning of Section 2, or we can express the stochastic differential equation in Stratonovich form, in which case we write

$$y_t = y_0 + \int_0^t V_0(y_\tau) d\tau + \sum_{i=1}^d \int_0^t V_i(y_\tau) \circ dW_\tau^i.$$

Two subtle notational changes can be spotted. First the ‘ $\circ dW_\tau^i$ ’ indicates that the stochastic integrals on the right are supposed to be interpreted in the Stratonovich sense; discussed presently. Second we now use the *Stratonovich drift vector field*  $V_0$  instead of the Itô drift vector field  $\tilde{V}_0$ . The relation between the two vector fields is

$$V_0 = \tilde{V}_0 - \frac{1}{2} \sum_{i=1}^d (V_i \cdot \partial_y V_i).$$

Importantly, when stochastic integrals are interpreted in the Itô sense, then they are limit of a left Riemann sum and when repeated integrals are computed, an Itô correction must be taken into account; a practical discussion of this point can be found in Higham [19, pp. 530–1]. For example, the correct evaluation of the an Itô repeated integral of a Wiener process with respect to itself is

$$\int_0^T W_\tau^i dW_\tau^i = \frac{1}{2} (W_T^i)^2 - \frac{1}{2} T.$$

Stratonovich integrals are interpreted as the limit of the midpoint rule, so for the corresponding Stratonovich integral the correct evaluation is

$$\int_0^T W_\tau^i \circ dW_\tau^i = \frac{1}{2} (W_T^i)^2.$$

Thus the rules of Stratonovich integral calculus match those of standard integral calculus. For this reason it is often preferable to use the Stratonovich representation for a stochastic differential equation. The two representations are equivalent, but it is important to know in which form you have been quoted the stochastic differential equation. Often this depends on the modeller and their field of interest. In finance applications the Itô representation predominates, and by simply replacing the given Itô drift vector field  $\tilde{V}_0$  by the *corresponding* Stratonovich drift vector field  $V_0$  above, one can proceed using standard integral calculus rules. In physical applications, the model is often often directly expressed in Stratonovich form. Hereafter we will use the Stratonovich representation and omit the ‘ $\circ$ ’ symbol, unless we specify otherwise.

## 3 Stochastic Taylor expansion

We follow the procedure we performed above for the ordinary differential equation to try to find the solution for the flow-map. In the process we obtain a solution series expansion called the stochastic Taylor series. We define the *flow-map*  $\varphi_t \in \text{Diff}(\mathcal{M})$  for the stochastic differential equation above as the map taking the solution configuration



on  $\mathcal{M}$  at time 0 to that at time  $t$ ; hence  $y_t = \varphi_t \circ y_0$ . Using the Stratonovich representation for a stochastic differential equation and the convention  $W_t^0 \equiv t$ , the chain rule for any function  $f \in \text{Diff}(\mathcal{M})$  yields the stochastic differential equation governing the evolution of  $f \circ y_t$  as follows

$$f \circ y_t = f \circ y_0 + \sum_{i=0}^d \int_0^t (V_i \cdot \partial_y f) \circ y_\tau dW_\tau^i.$$

As for the ordinary differential equation, setting  $f = \varphi_t$ , we can pull back the stochastic flow on  $\mathcal{M}$  to a functional stochastic differential equation on  $\text{Diff}(\mathcal{M})$  given by

$$\varphi_t = \text{id} + \sum_{i=0}^d \int_0^t V_i \circ \varphi_\tau dW_\tau^i.$$

To solve this equation, we set up the formal iterative procedure given by

$$\varphi_t^{(n+1)} = \text{id} + \sum_{i=0}^d \int_0^t V_i \circ \varphi_\tau^{(n)} dW_\tau^i,$$

with  $\varphi_t^{(0)} = \text{id}$ . By performing the iterations one can see formally, and prove rigorously, that the solution flow-map is given by the series expansion

$$\varphi_t = \text{id} + \sum_{i=0}^d (W_t^i) V_i + \sum_{i,j=0}^d \left( \int_0^t \int_0^{\tau_1} dW_{\tau_2}^i dW_{\tau_1}^j \right) V_{ij} + \dots.$$

Here we use the notation  $V_{ij} \equiv V_i \circ V_j$ . We can apply this to the initial data  $y_0 \in \mathcal{M}$  and obtain the *stochastic Taylor expansion* for the solution

$$y_t = y_0 + \sum_{i=0}^d (W_t^i) V_i(y_0) + \sum_{i,j=0}^d \left( \int_0^t \int_0^{\tau_1} dW_{\tau_2}^i dW_{\tau_1}^j \right) V_{ij}(y_0) + \dots.$$

We can express the solution series for the flow-map concisely as follows. Let  $\mathbb{A}^*$  denote the free monoid of words over the alphabet  $\mathbb{A} = \{0, 1, \dots, d\}$ . We adopt the standard notation for *Stratonovich integrals*, if  $w = a_1 \dots a_n$  then we set

$$J_w(t) := \int_0^t \dots \int_0^{\tau_{n-1}} dW_{\tau_n}^{a_1} \dots dW_{\tau_1}^{a_n}.$$

We also write the composition of the vector fields as  $V_w \equiv V_{a_1} \circ V_{a_2} \circ \dots \circ V_{a_n}$ . Then the flow-map is given by

$$\varphi_t = \sum_{w \in \mathbb{A}^*} J_w(t) V_w.$$

#### 4 PDE simulation

There is an intimate link between any stochastic differential equation and a prescribed *parabolic partial differential equation*. The link is given by the Feynman–Kac formula, which we give here in a very simple form. See for example Karlin and Taylor [21, pp. 222–4] for the full statement of the Feynman–Kac formula and its applications.

**Theorem 1 (Feynman–Kac formula)** *Consider the parabolic partial differential equation for  $t \in [0, T]$ :*

$$\partial_t u = \mathcal{L} u,$$

*with  $u(0, y) = f(y)$ . Here  $\mathcal{L} := V_0 + \frac{1}{2}(V_1^2 + \cdots + V_d^2)$  is a differential operator of order  $2N$ . Let  $y_t$  denote the solution to the stochastic differential equation for  $t \in [0, T]$ :*

$$y_t = y_0 + \int_0^t V_0(y_\tau) d\tau + \sum_{i=1}^d \int_0^t V_i(y_\tau) dW_\tau^i.$$

*Then, when  $y_0 = y$  we have:  $u(t, y) = \mathbf{E} f(y_t)$ .*

**Remark.** Note that using the relation between the Itô and Stratonovich drift vector fields, an equivalent formulation is  $\mathcal{L} \equiv \tilde{V}_0 \cdot \partial_y + \frac{1}{2} \sum_{i=1}^d (V_i \otimes V_i) : \partial_{yy}$ .

We provide a purely combinatorial proof of the Feynman–Kac formula. Before we begin, we need the following results, for the expectation of Stratonovich integrals and also a combinatorial expansion. Let  $\mathbb{D}^* \subset \mathbb{A}^*$  denote the free monoid of words constructed from the alphabet  $\mathbb{D} = \{0, 11, 22, \dots, dd\}$ . The expectation of a Stratonovich integral  $J_w$  is given by

$$\mathbf{E} J_w = \begin{cases} \frac{t^{n(w)}}{2^{d(w)} n(w)!}, & w \in \mathbb{D}^*; \\ 0, & w \in \mathbb{A}^* \setminus \mathbb{D}^*. \end{cases}$$

In the formula,  $d(w)$  is the number of non-zero consecutive pairs from  $\mathbb{D}$  in  $w$  and  $n(w) = z(w) + d(w)$ , where  $z(w)$  is the number of zeros in  $w$ . We also have the following combinatorial identity for all  $w \in \mathbb{D}^*$ :

$$(V_0 + \frac{1}{2}(V_1^2 + \cdots + V_d^2))^k \equiv \sum_{n(w)=k} (\frac{1}{2})^{d(w)} V_w,$$

where note that  $V_i^2 \equiv V_{ii}$ . In other words, expanding  $(V_0 + \frac{1}{2}(V_1^2 + \cdots + V_d^2))^k$  generates all the possible vector fields  $V_w$  with  $w \in \mathbb{D}^*$  and  $n(w) = k$ , with the appropriate coefficients of powers of one-half.

**Proof** In the series solution for the flow-map  $\varphi_t$ , all stochastic information is encoded in the words on the left and the geometric information on the right. Taking the expectation of the flow-map, noting that expectation is a linear operator, and using the two results

above for  $\mathbb{E} J_w$  and the combinatorial expansion, we get

$$\begin{aligned}
\mathbb{E} \varphi_t &= \mathbb{E} \sum_{w \in \mathbb{A}^*} J_w V_w \\
&= \sum_{w \in \mathbb{A}^*} (\mathbb{E} J_w) V_w, \\
&= \sum_{k \geq 0} \sum_{n(w)=k} \left(\frac{1}{2}\right)^{d(w)} \frac{t^k}{k!} V_w \\
&= \sum_{k \geq 0} \frac{t^k}{k!} (V_0 + \frac{1}{2}(V_1^2 + \dots + V_d^2))^k \\
&= \exp(t\mathcal{L}).
\end{aligned}$$

Now note that  $\exp(t\mathcal{L})$  generates the *semi-group* for the solution to the parabolic differential equation in the theorem.  $\square$

**Example (Heston model).** In the Heston model [18], a stock price  $S_t$  is modelled by a stochastic process  $x_t = \log S_t$  with variance process  $v_t$  which evolve according to:

$$\begin{aligned}
dx_t &= \mu dt + \sqrt{v_t} dW_t^1, \\
dv_t &= \kappa(\theta - v_t) dt + \varepsilon \sqrt{v_t} (\rho dW_t^1 + \sqrt{1 - \rho^2} dW_t^2),
\end{aligned}$$

given in Itô form. Note that the variance is a mean-reverting process; it tries to revert to the mean value  $\theta$  at rate  $\kappa$ . Using the Feynman–Kac formula the corresponding partial differential equation for  $u(x, v, t) = \mathbb{E}(f(x_t, v_t) \mid x_0 = x, v_0 = v)$  is

$$u_t = \mu u_x + \kappa(\theta - v) u_v + \frac{1}{2} v u_{xx} + \rho \varepsilon v u_{xv} + \frac{1}{2} \varepsilon^2 v u_{vv}.$$

**Remark.** The Feynman–Kac formula shows how we can solve a partial differential equation to obtain information about the solution  $y_t$  to the stochastic differential equation at time  $t \in [0, T]$ . In the reverse direction, to numerically solve high dimensional diffusion problems in the form of deterministic partial differential equations, we need only simulate an  $N$ -dimensional stochastic equation for  $y_t$  and then compute the expectation  $\mathbb{E} y_t$  to find the solution.

## 5 Monte–Carlo simulation

In *Monte–Carlo simulation*, we generate a set of suitable multidimensional sample paths say  $\hat{W}(\omega) := (\hat{W}^1(\omega), \dots, \hat{W}^d(\omega))$  on  $[0, T]$ . We generate a large finite set of paths, each labelled by  $\omega$ . Here the number of paths, say  $P$ , must be large enough so that, for example, any statistical information for the solution  $y_t$  that we want to extract is sufficiently robust. For each sample path  $\hat{W}(\omega)$ , we generate a sample path solution  $\hat{y}(\omega)$  to the stochastic differential equation on  $[0, T]$ . This can often only be achieved approximately, by using a truncation of the stochastic Taylor expansion for the solution  $y$  on successive small subintervals of  $[0, T]$ . Having generated a set of

approximate solutions  $\hat{y}_t(\omega_i)$  at time  $t \in [0, T]$  for every  $\omega_i$  for  $i = 1, \dots, P$ , we estimate the expectation  $\mathbb{E} f(\hat{y}_t)$  by computing

$$\frac{1}{P} \sum_{i=1}^P f(\hat{y}_t(\omega_i))$$

regarded as a suitable approximation for

$$\mathbb{E} f(y_t) := \int f(y_t(\omega)) dP(\omega)$$

over all possible paths. Now a natural question arises. Do the suitable paths  $\hat{W}(\omega)$  we generate, have to be sample Brownian paths to compute the mean above, or can we choose different paths that will still generate the expectation effectively? We discuss the latter case (weak simulation) briefly next. We then move onto a more comprehensive treatment of the former (strong simulation) case, which also allows us to include path-dependent features.

### 5.1 Weak simulation

There are several *weak simulation* strategies to approximate  $\mathbb{E} f(\hat{y}_t)$ . The simplest and most common is to replace the driving Wiener paths  $W^i$  by paths generated as follows. Construct paths by generating increments  $\Delta W^i(t_n, t_{n+1})$  over the computation subinterval  $[t_n, t_{n+1}]$  by the binomial branching process

$$P(\Delta W^i(t_n, t_{n+1}) = \pm\sqrt{h}) = \frac{1}{2},$$

where  $h = t_{n+1} - t_n$ . Then, depending on the ordinary numerical integration scheme employed for each such path, one can show that for some order of convergence  $p$ , for some  $t \in [0, T]$ , we have

$$\|\mathbb{E} f(y_t) - \mathbb{E} f(\hat{y}_t)\| = \mathcal{O}(h^p).$$

See Kloeden and Platen [22] for more details. Another promising method is the cubature method of Lyons and Victoir [28].

### 5.2 Strong simulation

In a *strong simulation*, discrete increments  $\Delta W^i$  in each computation interval are directly sampled from the Gaussian distribution. Indeed we generate each multidimensional path  $\hat{W}(\omega)$  by choosing increments in each component as follows

$$\Delta W^i(t_n, t_{n+1}) \sim \sqrt{h} \cdot \mathbf{N}(0, 1).$$

This is more expensive, but sample paths  $\hat{W}(\omega)$  generated in this way, allow us to compare  $\hat{y}_t(\omega)$  and  $y_t(\omega)$  directly in the sense that one can show

$$\mathbb{E} \|y_t - \hat{y}_t\| = \mathcal{O}(h^{\frac{p}{2}})$$

for some order of strong convergence  $p/2$ ; which we will discuss in detail in Section 7. Often in practice we take  $\|\cdot\|$  to be the Euclidean norm so that the convergence shown is in the  $L^2$ -norm.

Given a sample multidimensional path  $\hat{W}(\omega)$  on  $[0, T]$ , how do we actually construct an approximate solution  $\hat{y}_t$ ? Here we are guided by the stochastic Taylor expansion. Indeed, classical strong numerical methods are based on truncating the stochastic Taylor expansion

$$y_t = \sum_{w \in \mathbb{A}^*} J_w(t) V_w(y_0),$$

and applying the approximation over successive subintervals of the global interval of integration  $[0, T]$ ; see Kloeden and Platen [22] or Milstein [31]. We present three simple example numerical approximation methods.

### 5.3 Euler–Maruyama method

If we truncate the Itô form of the stochastic Taylor series after the first order terms we generate the *Euler–Maruyama numerical method* as follows:

$$\hat{y}_{n+1} = \hat{y}_n + h \tilde{V}_0(\hat{y}_n) + \sum_{i=1}^d (\Delta W^i(t_n, t_{n+1})) V_i(\hat{y}_n).$$

This is a numerical scheme with global order of convergence of  $h^{\frac{1}{2}}$ . We explain in Section 7 why we have used the Itô drift vector field here.

### 5.4 Milstein method

We now truncate the stochastic Taylor series after the second order terms. This generates the *Milstein numerical method* given by

$$\hat{y}_{n+1} = \hat{y}_n + h V_0(\hat{y}_n) + \sum_{i=1}^d (\Delta W^i(t_n, t_{n+1})) V_i(\hat{y}_n) + \sum_{i,j=1}^d J_{ij}(t_n, t_{n+1}) V_{ij}(\hat{y}_n).$$

An important and expensive ingredient in this method, is the simulation of the multiple integrals  $J_{ij}(t_n, t_{n+1})$  for  $i \neq j$  shown, on each integration step. When  $i = j$ , the multiple integrals  $J_{ii}(t_n, t_{n+1})$  are cheaply evaluated by

$$J_{ii}(t_n, t_{n+1}) = \int_{t_n}^{t_{n+1}} \int_{t_n}^{\tau_1} dW_{\tau_2}^i dW_{\tau_1}^j = \frac{1}{2} (\Delta W^i(t_n, t_{n+1}))^2.$$

When  $i \neq j$  we have by integration by parts that

$$J_{ji} = J_i J_j - J_{ij}.$$

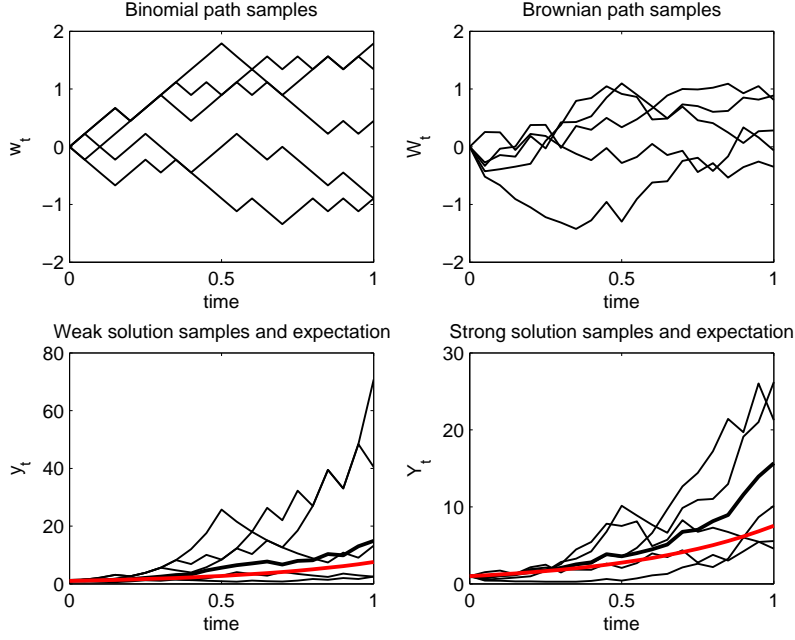
Hence we need only compute one double integral for each pair  $i \neq j$ . Equivalently we need only compute the Lévy area given by

$$A_{ij} := \frac{1}{2} (J_{ij} - J_{ji}),$$

since  $J_{ij} = \frac{1}{2} J_i J_j + A_{ij}$ . By Stokes' Theorem, the Lévy area on the interval  $[t_n, t_{n+1}]$  is the chordal area for the path  $(W^i, W^j)$  on  $[t_n, t_{n+1}]$ . This can be observed directly by definition since

$$A_{ij}(t) = \frac{1}{2} \int_0^t W_{\tau}^i dW_{\tau}^j - W_{\tau}^j dW_{\tau}^i.$$

We consider the issue of simulating the Lévy area in some detail in Section 6. The Milstein scheme has global order of convergence  $h$ .



**Fig. 2** Weak and strong simulations for the scalar linear example given in Section 2, with  $a = 3$  and  $b = 2$ . For this example we took  $y_0 = 1$ ,  $T = 1$ ,  $h = 0.05$  and  $P = 10$  for both simulations, though only 5 sample paths are shown above in all cases. Top left are 5 sample binomial branching paths  $w$ , and top right are 5 sample Brownian paths  $W$ . Lower left are 5 sample solution paths  $y$  using the binomial branching paths, while lower right are 5 sample solution paths  $Y$  using the Brownian paths; both computed using the Euler–Maruyama method. At each time-step the thick black line shows the average value over  $P = 10$  samples and the red line is the analytic solution for the expectation.

### 5.5 Castell–Gaines method

Consider the exponential Lie series  $\psi_t = \log \varphi_t$  generated by taking the logarithm of the stochastic Taylor series for the flow-map, i.e.

$$\begin{aligned} \psi_t &= (\varphi_t - \text{id}) - \frac{1}{2}(\varphi_t - \text{id})^2 + \frac{1}{3}(\varphi_t - \text{id})^3 + \cdots \\ &= \sum_{i=0}^d J_i V_i + \sum_{i>j} \frac{1}{2}(J_{ij} - J_{ji})[V_i, V_j] + \cdots. \end{aligned}$$

This series is also known as the Chen–Strichartz, Chen–Fleiss or Magnus series. The Castell–Gaines method is a strong numerical method based on truncating the exponential Lie series. As for the methods above, we generate a set of multidimensional paths  $\hat{W}(\omega)$  on  $[0, T]$  with Wiener increments  $\Delta W^i(t_n, t_{n+1})$  sampled on the scale  $h = t_{n+1} - t_n$ . On each computation interval  $[t_n, t_{n+1}]$ , we replace the  $J_i(t_n, t_{n+1})$  by the Normal samples  $\Delta W^i(t_n, t_{n+1})$ . If required, the Lévy area increments  $A_{ij}(t_n, t_{n+1})$  shown, are also replaced by suitable samples  $\hat{A}_{ij}(t_n, t_{n+1})$  as we outline in Section 6.

Then across the computation interval  $[t_n, t_{n+1}]$ , we have

$$\hat{\psi}_{t_n, t_{n+1}} = \sum_{i=0}^d (\Delta W^i(t_n, t_{n+1})) V_i + \sum_{i>j} \hat{A}_{ij}(t_n, t_{n+1}) [V_i, V_j].$$

The solution at time  $t_{n+1}$  is then approximately given by

$$\hat{y}_{t_{n+1}} \approx \exp(\hat{\psi}_{t_n, t_{n+1}}) \circ \hat{y}_{t_n}.$$

Note that for each path  $\Delta W^i(t_n, t_{n+1})$  and  $\hat{A}_{ij}(t_n, t_{n+1})$  are fixed constants. Hence the truncated Lie series  $\hat{\psi}_{t_n, t_{n+1}}$  is itself an autonomous vector field. Thus, for  $\tau \in [0, 1]$  and with  $u(0) = \hat{y}_{t_n}$ , we solve the ordinary differential equation

$$u'(\tau) = \hat{\psi}_{t_n, t_{n+1}} \circ u(\tau).$$

Using a suitable high order ordinary differential integrator generates  $u(1) \approx \hat{y}_{t_{n+1}}$ .

Without the Lévy area the Castell–Gaines method has global order of convergence  $h^{\frac{1}{2}}$ ; while with the Lévy area it has global order of convergence  $h$ . Castell and Gaines [10, 11] prove that their strong order  $h^{\frac{1}{2}}$  method is always more accurate than the Euler–Maruyama method. Indeed they prove that this method is *asymptotically efficient* in the sense of Newton [34]. Further in the case of a single driving Wiener process ( $d = 1$ ), they prove the same is true for their strong order  $h$  method. By asymptotically efficient we mean, quoting from Newton, that they “minimize the leading coefficient in the expansion of mean-square errors as power series in the sample step size”.

## 6 Simulating the Lévy area

A fundamental and crucial aspect to the implementation of strong order one or higher integrators for stochastic differential equations, is the need to successfully simulate the Lévy chordal areas  $A_{ij}(t_n, t_{n+1})$ , when the diffusion vector fields do not commute. This aspect is more than just a new additional concern once we step off the cliff edge of simple path increment approximations with frozen vector fields characterized by the Euler–Maruyama approximation. It also represents a substantial technical difficulty. Here we will outline several methods employed to simulate it sufficiently accurately; the important distinguishing criterion for the success of the method will be its asymptotic rate of convergence as  $h \rightarrow 0$ . A sample survey of these methods can be found in Rydén and Wiktorsson [36]. Here we will focus on the case of two independent Wiener processes  $W^1$  and  $W^2$  and the requirement to simulate  $A_{12}(h) := A_{12}(t_n, t_{n+1})$ , given the Normal increments  $\Delta W^1(h) := \Delta W^1(t_n, t_{n+1})$  and  $\Delta W^2(h) := \Delta W^2(t_n, t_{n+1})$  across  $[t_n, t_{n+1}]$ .

### 6.1 Simulating Normal random variables

We will start with the question: what is the most efficient method for generating  $\Delta W^1(h)$  and  $\Delta W^2(h)$ ? The simple and direct answer is to use the Matlab command

`sqrt(h)*randn`

This command invokes an algorithm that has been scrupulously refined and adapted over the years. One of the simplest efficient earlier incarnations of this algorithm is Marsaglia's polar method [30]. The Box–Müller method is also very simple but not quite as efficient; see Kloeden and Platen [22] for a discussion of these issues. Also see Moro's inversion method [33]. We outline Marsaglia's method here because of its simplicity and effectiveness.

**Algorithm 1 (Marsaglia's method)** To produce two standard Normal samples:

1. Generate two independent uniform random samples  $U_1, U_2 \in \text{Unif}([-1, 1])$ ;
2. If  $S := U_1^2 + U_2^2 < 1$  continue, otherwise repeat Step 1;
3. Compute  $X_i = U_i / \sqrt{-2 \ln(S)/S}$ , for  $i = 1, 2$ ; then  $X_1$  and  $X_2$  are independent standard Normal samples.

## 6.2 Conditional distribution of Lévy area

The *characteristic function*  $\hat{\phi}$  of the probability density function for  $\xi = A_{12}(h)$  given  $\Delta W^1(h)$  and  $\Delta W^2(h)$  is

$$\hat{\phi}(\xi) = \frac{\frac{1}{2}h\xi}{\sinh(\frac{1}{2}h\xi)} \exp\left(-\frac{1}{2}a^2\left(\frac{1}{2}h\xi \coth(\frac{1}{2}h\xi) - 1\right)\right)$$

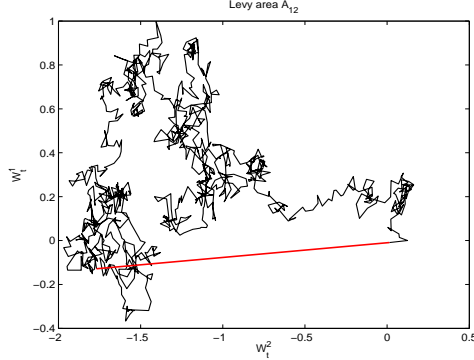
where  $a^2 = ((\Delta W^1(h))^2 + (\Delta W^2(h))^2)/h$ . Lévy derived this in a very succinct calculation in 1951; see Lévy [24, pp. 171–3]. Since  $\hat{\phi}$  is the characteristic function, i.e. the Fourier transform of the corresponding probability density function, the actual probability density function  $\phi$  is given by the inverse Fourier transform (see for example Gaines and Lyons [13]):

$$\phi(x) = \frac{1}{\pi} \int_0^\infty \hat{\phi}(\xi) \cos(x\xi) d\xi.$$

The ungainly form of this probability density function means that generating samples is not likely to be easy. For example, the simplest method for sampling from a continuous distribution  $f$  is based on the inversion of its (cumulative) distribution function  $F(x) := \int_{-\infty}^x f(\eta) d\eta$ . If we sample from the uniform distribution, say  $U \sim \text{Unif}([0, 1])$ , then  $F^{-1}(U)$  is a sample from the target distribution. For this to be a practical sampling method we must have an analytic form for  $F$  or an extremely efficient quadrature approximation for the integral in  $F$  at our disposal. We don't have this for the probability density function of the Lévy area  $\phi$ .

Several methods have been proposed for sampling from  $\phi$ . Gaines and Lyons [13] proposed one of the most efficient, based on Marsaglia's rectangle–wedge–tail method. However it can be complicated to implement. Kloeden and Platen [22] and Wiktorsson have proposed methods based on the Karhunen–Loève expansion are much easier to code. Rydén and Wiktorsson [36] proposed a method based on recognising the characteristic function  $\hat{\phi}$  as a product of characteristic functions for a logistic random variable and an infinite sum of Poisson mixed Laplace random variables. Gaines and Lyons [14] also proposed a method based on the conditional expectation of the Lévy area, conditioned on intermediate Wiener increments. We discuss these methods in the following four sections. Stump and Hill [39] have also proposed a very efficient method, whose potential in a practical implementation is yet to be explored.





**Fig. 3** Sample two-dimensional Wiener path and enclosed chordal Lévy area.

### 6.3 Karhunen–Loève expansion method

Lévy [24] derived the form for the characteristic function  $\phi$  for the Lévy area, using the Karhunen–Loève expansion for a Brownian bridge. This is an expansion in orthogonal basis functions. The details can be found in Lévy [24] or Kloeden and Platen [22]. If  $U_k, V_k, X_k, Y_k$  are independent  $N(0, 1)$  samples, also independent of  $\Delta W^1(h)$  and  $\Delta W^2(h)$ , then the Lévy area can be represented by

$$A_{12}(h) = \frac{h}{2\pi} \sum_{k=1}^{\infty} \frac{1}{k} \left( U_k (Y_k - \sqrt{\frac{2}{h}} \Delta W^2(h)) - V_k (X_k - \sqrt{\frac{2}{h}} \Delta W^1(h)) \right).$$

In practice, we truncate this expansion to only include  $k \leq Q$  terms and use the truncation,  $\hat{A}_{12}(h)$ , as an approximation for the Lévy area. The important question now, is as far as strong simulation is concerned, how many standard Normal random variables do we need to simulate in order to have a sufficiently accurate Lévy area sample  $\hat{A}_{12}(h)$ , i.e. how large must  $Q$  be? Note that the coefficients in the above expansion scale like  $h/k$ . The properties of the tail of the series, i.e. all the terms for  $k \geq Q + 1$ , mean that it scales as  $h/\sqrt{Q}$ . For a Milstein numerical approximation we require that the strong error is locally of order  $h^{\frac{3}{2}}$ ; see Section 7. Hence we must choose  $Q \approx h^{-1}$  for a sufficiently accurate sample.

### 6.4 Rydén and Wiktorsson's method

Rydén and Wiktorsson [36] proposed several methods, we detail here the most expedient. The characteristic function  $\hat{\phi}$  is the product of two characteristic functions

$$\hat{\phi}_{X(h)}(\xi) = \frac{\frac{1}{2}h\xi}{\sinh(\frac{1}{2}h\xi)} \quad \text{and} \quad \hat{\phi}_{Y(h)}(\xi) = \exp\left(-\frac{1}{2}a^2\left(\frac{1}{2}h\xi \coth(\frac{1}{2}h\xi) - 1\right)\right)$$

corresponding to the random variables  $X(h)$  and  $Y(h)$ , respectively. We observe that  $\hat{\phi}_{X(h)}$  is the characteristic function of a logistic random variable which can be generated

by the inverse method, i.e. pick  $U \sim \text{Unif}([0, 1])$  and let  $X(h) = (h/2\pi) \log(U/(1-U))$ . Then using the identity

$$z \coth z - 1 = 2 \sum_{k=1}^{\infty} \frac{z^2}{\pi^2 k^2 + z^2},$$

we observe that

$$\hat{\phi}_{Y(h)}(\xi) = \exp\left(-a^2 \sum_{k=1}^{\infty} \frac{\xi^2}{(2\pi k/h)^2 + \xi^2}\right).$$

This can be viewed as a sum of compound Poisson random variables. Indeed if for each  $k \in \mathbb{N}$ , we generate  $N_k \sim \text{Poisson}(a^2)$  and then, for  $j = 1, \dots, N_k$  generate independent Laplace random variables  $Y_{jk} \sim \text{Laplace}(1/k)$ , then

$$Y(h) = \frac{h}{2\pi} \sum_{k=1}^{\infty} \sum_{j=1}^{N_k} Y_{jk},$$

has density  $\phi_{Y(h)}$ . In a practical implementation we truncate this expansion to include  $k \leq Q$  terms, and use the truncation as an approximation for the Lévy area. Further the tail sum, by the central limit theorem, is asymptotically Normally distributed and can be approximated by a Normal random variable. This provides quite a dramatic improvement as it is possible to show that this method only requires the number of standard Normal samples to be  $Q \approx h^{-\frac{1}{2}}$ .

### 6.5 Wiktorsson's method

Wiktorsson proposed a method that uses the Karhunen–Loève expansion method, but also simulates the tail sum as in the last method. Again, by the central limit theorem, the tail sum can be approximated by a Normal random variable, and the corresponding improvement is that this method only requires the number of standard Normal samples to be  $Q \approx h^{-\frac{1}{2}}$ . Wiktorsson's method has been successfully implemented by Gilling and Shardlow [16] in their SDELab, to where the interested reader is referred.

### 6.6 Conditional expectation

One more approach to simulating the Lévy area, or equivalently  $J_{12}(t_n, t_{n+1})$ , is based on replacing  $J_{12}(t_n, t_{n+1})$ , by its conditional expectation  $\hat{J}_{12}(t_n, t_{n+1})$ , as follows. Suppose we are about to perform the numerical update for the solution across the interval  $[t_n, t_{n+1}]$ . We generate  $Q$  pairs of independent standard Normal random variables  $X_q, Y_q \sim N(0, 1)$  for  $q = 1, \dots, Q$ . Set  $\tau_q = t_n + q\Delta t$ , for  $q = 0, \dots, Q-1$ , and  $\Delta W^1(\tau_q) = \sqrt{\Delta t} X_q$  and  $\Delta W^2(\tau_q) = \sqrt{\Delta t} Y_q$ , where  $\Delta t$  is defined by  $Q\Delta t = h$ . We thus generate a two-dimensional Brownian sample path on  $[t_n, t_{n+1}]$ . We can take  $\Delta W^1(h)$  and  $\Delta W^2(h)$  to be the increments across the interval  $[t_n, t_{n+1}]$ . More importantly, we can use the intervening path information we have generated on the scale  $\Delta t$

to approximate  $J_{12}(t_n, t_{n+1})$ . Indeed,  $J_{12}(t_n, t_{n+1})$  can be expressed as

$$\begin{aligned} J_{12}(t_n, t_{n+1}) &= \int_{t_n}^{t_{n+1}} \int_{t_n}^{\tau} dW_{\tau_1}^1 dW_{\tau}^2 \\ &= \sum_{q=0}^{Q-1} \int_{\tau_q}^{\tau_{q+1}} (W_{\tau}^1 - W_{\tau_q}^1) + (W_{\tau_q}^1 - W_{t_n}^1) dW_{\tau}^2 \\ &= \sum_{q=0}^{Q-1} J_{12}(\tau_q, \tau_{q+1}) + \sum_{q=0}^{Q-1} (W_{\tau_q}^1 - W_{t_n}^1) \Delta W^2(\tau_q). \end{aligned}$$

The quantity

$$\hat{J}_{12}(t_n, t_{n+1}) := \sum_{q=0}^{Q-1} (W_{\tau_q}^1 - W_{t_n}^1) \Delta W^2(\tau_q)$$

represents the expectation of  $J_{12}(t_n, t_{n+1})$  conditioned on the increments  $\Delta W^1(\tau_q)$  and  $\Delta W^2(\tau_q)$ . From an algebraic and geometric perspective,  $\hat{J}_{12}(t_n, t_{n+1})$  represents a suitable approximation to  $J_{12}(t_n, t_{n+1})$ . Computing its mean-square strong error we see that

$$\|J_{12}(t_n, t_{n+1}) - \hat{J}_{12}(t_n, t_{n+1})\|_{L^2}^2 = \sum_{q=0}^{Q-1} \|J_{12}(\tau_q, \tau_{q+1})\|_{L^2}^2 = Q(\Delta t)^2 = h^2/Q.$$

Hence its root-mean-square strong error is  $h/\sqrt{Q}$ . Thus, as for the Karhunen–Loève expansion approach, to achieve a suitable approximate sample for the stochastic area integral, this method requires  $Q \approx h^{-1}$ . One advantage of this method is that it is very convenient for generating log-log error plots.

## 7 Strong error

We will focus here on the global, strong  $L^2$  error. In practical terms, the global error is generated by the accumulation of contributions from the local error. The local error is itself the leading order terms in the remainder  $R_t$ , say, of our truncated stochastic Taylor series. Note that there is also a contribution to the global error from the approximate simulation of the Lévy area, however we will assume here, that the Lévy area has been sufficiently accurately simulated, as discussed in detail in the last section, so that its contribution is *small*, in comparison to the truncation error.

Suppose we base a strong numerical approximation on truncating the stochastic Taylor expansion (in Stratonovich form). Let  $\hat{y}$  denoted the truncated expansion and  $R$  the corresponding remainder; hence the exact solution is  $y = \hat{y} + R$ . To guarantee our numerical scheme based on such a truncation is globally of order  $h^m$ , where  $m \in \mathbb{Z}/2$ , which terms must we keep in  $\hat{y}$ ? We give the following rule.

**Rule of Thumb:** Terms in the remainder  $R$  of  $L^2$  measure  $h^m$  with:

- zero expectation, accumulate so as to contribute to the global error as  $h^{m-\frac{1}{2}}$  order terms;
- non-zero expectation, accumulate so as to contribute to the global error as  $h^{m-1}$  order terms.

Hence to achieve an integrator with global error of order  $h^m$ , we must retain in  $\hat{y}$ :

- all terms with  $L^2$  measure of order  $h^{m'}$  for all  $m' \leq m$ ;
- the expectation of all terms of order  $h^{m+\frac{1}{2}}$  which have non-zero expectation (the corresponding terms left in remainder will then have zero expectation).

**Example (Euler–Maruyama).** Recall that we based the Euler–Maruyama approximation on the truncated Itô Taylor series. If we had truncated the stochastic Taylor series in Stratonovich form, then according to the rules above, to achieve global order  $h^{\frac{1}{2}}$  we should retain in our integrator the expectation of the terms

$$\sum_{i=1}^d (V_i \cdot \partial_y V_i) J_{ii}.$$

Since  $\mathbb{E} J_{ii} = \frac{1}{2}h$ , we thus recover the corresponding truncated Itô Taylor series.

## 8 Further issues

There are many important simulation issues we have not had space to discuss. Chief among these is the numerical stability of the strong methods we have explicitly outlined. This issue is discussed in Higham [19] and more can be found for example in Buckwar, Horváth–Bokor and Winkler [8].

## A Stratonovich to Itô relations

We give here some Stratonovich to Itô relations for convenience for the reader—more details can be found in Kloeden and Platen [22]. For the words  $w$  shown, the Stratonovich integrals  $J_w$  can be expressed in terms of Itô integrals  $I_w$  as follows:

$$\begin{aligned} w = a_1 a_2: J_w &= I_w + \frac{1}{2} I_0 \delta_{a_1=a_2 \neq 0}; \\ w = a_1 a_2 a_3: J_w &= I_w + \frac{1}{2} (I_{0a_3} \delta_{a_1=a_2 \neq 0} + I_{a_1 0} \delta_{a_2=a_3 \neq 0}); \\ w = a_1 a_2 a_3 a_4: J_w &= I_w + \frac{1}{4} I_{00} \delta_{a_1=a_2 \neq 0} \delta_{a_3=a_4 \neq 0} \\ &\quad + \frac{1}{2} (I_{0a_3 a_4} \delta_{a_1=a_2 \neq 0} + I_{a_1 0 a_4} \delta_{a_2=a_3 \neq 0} + I_{a_1 a_2 0} \delta_{a_3=a_4 \neq 0}). \end{aligned}$$

Note that the expectation of any Itô integral  $I_w$  is zero, i.e.:  $\mathbb{E} I_w = 0$  for any word  $w \in \mathbb{A}^*$  which has at least one non-zero letter.

## B Sample program for weak and strong Euler–Maruyama

We provide the listing for the weak vs strong Euler–Maruyama simulation shown in Figure 2.

**Listing 1** Weak vs strong simulation

---

```
%%      Weak and strong simulation example plot
%%      Euler-Maruyama approximations
%%      Example scalar linear SDE

%%      Parameter values
```

---

```

a=3.0; % drift coefficient
b=1.4; % diffusion coefficient is b
y0=1.0; % initial data

P=10; % total # of sample paths
h=0.05; % stepsize
T=1.0; % global time interval
N=T/h; % number of subintervals

%% Binomial branching process increments

dw=zeros(N,P);
w=zeros(N+1,P);
binom=binornd(1,1/2,[N,P]); % Gives 0 or 1 with prob 1/2
dw=sqrt(h)*(1-2*binom); % Binomial increments dw
w(2:N+1,:)=cumsum(dw,1); % Binomial paths themselves

%% Weak solution by Euler-Maruyama approximation

y=zeros(N+1,P); % y is weak solution

for p=1:P % set loop for each path
    y(1,p)=y0; % initial data

    for n=1:N
        y(n+1,p)=y(n,p)+a*y(n,p)*h+b*y(n,p)*dw(n,p);
    end
end

%% Approximate Wiener path increments

dW=zeros(N,P);
W=zeros(N+1,P);
dW=sqrt(h)*randn(N,P); % Brownian increments dW
W(2:N+1,:)=cumsum(dW,1); % Brownian paths themselves

%% Strong solution by Euler-Maruyama approximation

Y=zeros(N+1,P); % Y is strong solution

for p=1:P % set loop for each path
    Y(1,p)=y0; % initial data

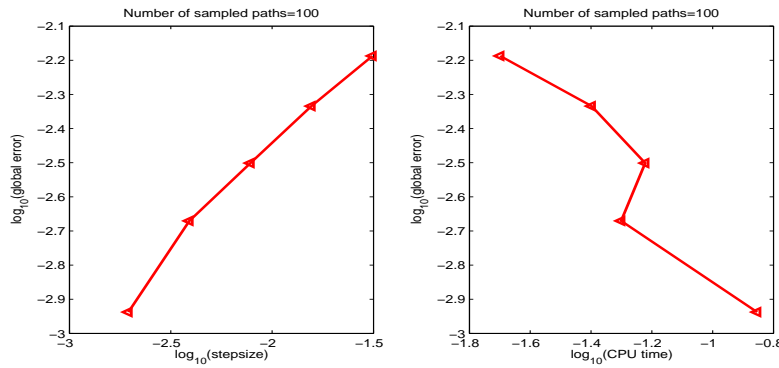
    for n=1:N
        Y(n+1,p)=Y(n,p)+a*Y(n,p)*h+b*Y(n,p)*dW(n,p);
    end
end

%% Compute the expectations of y and Y at each timestep

expect_y=zeros(N+1,1);
expect_Y=zeros(N+1,1);

for n=1:N+1
    expect_y(n)=mean(y(n,:));
    expect_Y(n)=mean(Y(n,:));
end

```



**Fig. 4** Error vs stepsize and error vs CPU time for the Heston model. The parameter values can be seen in the program listing.

## C Example strong simulation program

### C.1 Heston model strong simulation

We provide here a sample program that shows how to perform log-log error plots for a strong simulation. We used a real example, the Heston model, and applied the full truncation Euler–Maruyama type numerical scheme devised by Lord, Koekkoek and Van Dijk [25]. The log-log error vs stepsize, and error vs CPU time, are shown in Fig. 4. Note that to estimate the strong global error, we must compare solutions for different stepsizes along the *same* path, before taking the expectation.

**Listing 2** Heston model strong simulation

```
%%          Heston model integration

% Global data

T0=0;                % integration interval...
T=1;                 % ...is [T0,T]
M=10;
hmin=(T-T0)/2^M;     % smallest timestep
Mstart=4;
hmax=(T-T0)/2^Mstart; % largest timestep
Q=(1/(hmin))*(hmax/hmin); % # of quad pts for hmin
dt=hmax/Q;           % quad scale for hmin
R=M-Mstart+1;        % # of solution approximations...
                        % ...at different stepsizes
P=100;               % total # of Brownian paths
alpha=2.0;            % parameters
theta=0.09;
beta=0.1;
rho=0.5;
mu=0.05;
ic=[1.0; 0.09];     % initial data

YFT=zeros(P,R,2);    % approximate solution
clockYFT=zeros(1,R); % CPU timings
```

---

```

for p=1:P
    for r=1:R
        YFT(p,r,:)=ic;           % make sure start with IC
    end
end

for jj=1:2^Mstart
    YFTold=YFT;

    for p=1:P                     % loop for each path

        %%           Start loop computing over intervals of length hmax

        siv=hmax/hmin;

        %%           Generate dW1 and dW2 on smallest scale

        dW1=sqrt(dt)*randn(1,Q);   % Brownian increments dW1
        dW2=sqrt(dt)*randn(1,Q);   % Brownian increments dW2
        dW0=(zeros(1,Q)+1)*dt;

        %%           Start loop computing Js with different stepsizes

        for r=1:R
            SF=2^(r-1);           % scale factor for stepsizes...
            h=SF*hmin;            % stepsize
            L=hmax/h;             % # of timesteps needed
            QR=Q/(SF^2);          % # of quadrature steps (total)...

            % Compute dw0, dw1, dw2, w0, w1, w2

            dw0=zeros(1,QR);
            dw1=zeros(1,QR);
            dw2=zeros(1,QR);
            w0=zeros(1,QR);
            w1=zeros(1,QR);
            w2=zeros(1,QR);

            for j=1:QR
                dw0(j)=sum(dw0((j-1)*(SF^2)+1:j*(SF^2)));
                dw1(j)=sum(dw1((j-1)*(SF^2)+1:j*(SF^2)));
                dw2(j)=sum(dw2((j-1)*(SF^2)+1:j*(SF^2)));
            end

            QF=QR/L;               % # of quadrature steps in h...

            for n=1:L
                w0((n-1)*QF+1:n*QF)=cumsum(dw0((n-1)*QF+1:n*QF));
                w1((n-1)*QF+1:n*QF)=cumsum(dw1((n-1)*QF+1:n*QF));
                w2((n-1)*QF+1:n*QF)=cumsum(dw2((n-1)*QF+1:n*QF));
            end

            oldclockYFT=clockYFT(r);
            ts=cputime;
            YFT(p,r,:)=YFTapprox(p,h,L,T0,QF,dw0,dw1,dw2,w0,w1,w2, ...
                                alpha,theta,beta,rho,mu,...
                                YFTold(p,r,:));
            clockYFT(r)=cputime-ts+oldclockYFT;
        end
    end
end

```

```

end
end

stepsizes=log10((2.^([1:R-1]))*hmin);

save('stepsizes','stepsizes')
save('P','P')
save('YFT','YFT')
save('clockYFT','clockYFT')

```

## C.2 Program listing: integrator

We give the program listing for the Heston model full truncation Euler–Maruyama integrator.

### Listing 3 Full truncation Euler–Maruyama integrator

---

```

%%      Full truncation for volatility and exponential for index:

function trunc=YFTapprox(p,h,L,T0,QR,dW0,dW1,dW2,...
                        W0,W1,W2,alpha,theta,beta,rho,mu,ic)

trunc=zeros(2,1);

% Compute dJ1, dJ2 on scale h=SF*hmin

J1=zeros(1,L);
J2=zeros(1,L);
J1(1)=W1(QR);
J2(1)=W2(QR);
pts=2*QR:QR:L*QR;
J1(2:L)=W1(pts);
J2(2:L)=W2(pts);

%%      initial data

S=ic(1);                                % asset price
v=ic(2);                                % volatility

for n=1:L
    Sold=S;
    vold=v;
    S=exp((mu-max(0,vold)/2)*h+sqrt(max(0,vold))*J1(n))*Sold;
    v=vold+alpha*(theta-max(0,vold))*h+beta*(rho*J1(n)...
        +sqrt(1-rho^2)*J2(n))*sqrt(max(0,vold));
end

trunc=[S; v];

end

```

## C.3 Program listing: log-log strong error plots

The following program performs the log-log plots for the strong  $L^2$  error measure.



**Listing 4** Log-log strong error plots

---

```

%%      Log-log strong error plots

load stepsizes
load YFT
load clockYFT
load P

R=length(stepsizes);
errorYFT=zeros(1,R);
diffYFT=zeros(P,R);

% the "norm" below is the Euclidean norm

for r=1:R
    for p=1:P
        diffYFT(p,r)=norm(YFT(p,r+1)-YFT(p,1));
    end
end

% now take the L^2 norm measure

errorYFT=sqrt(mean(diffYFT.^2,1));

figure
subplot(1,2,1)
plot(stepsizes(1:end-1),log10(errorYFT(1:end-1)),...
      '-ks','LineWidth',2)
xlabel('log_{10}(stepsize)')
ylabel('log_{10}(global_error)')
title(['Number_of_sampled_paths=',int2str(P)])
subplot(1,2,2)
plot(log10(clockYFT(2:end-1)),log10(errorYFT(1:end-1)),...
      '-ks','LineWidth',2)
xlabel('log_{10}(CPU_time)')
ylabel('log_{10}(global_error)')
title(['Number_of_sampled_paths=',int2str(P)])

```

**References**

1. Arnold, L.: Stochastic differential equations. Wiley, 1974.
2. Azencott, R.: Formule de Taylor stochastique et développement asymptotique d'intégrales de Feynman, Seminar on Probability XVI, Lecture Notes in Math., 921 (1982), Springer, pp. 237–285.
3. Baudoin, F.: An introduction to the geometry of stochastic flows. Imperial College Press, 2004.
4. Baudoin, F.: Introduction to the geometry of stochastic differential equations and diffusion semigroups, SNSB Bucharest lectures, 2009.
5. Ben Arous, G.: Flots et series de Taylor stochastiques, Probab. Theory Related Fields, 81 (1989), pp. 29–77.
6. Boyle, P., Broadie, M., Glasserman, P.: Monte Carlo methods for security pricing, Journal of Economic Dynamics and Control, 21(8–9) (1997) pp. 1267–1321.
7. Brown, D. L., Bell, J., Estep, D., Gropp, W., Hendricksen, B., Keller–McNulty, S., Keyes, D., Oden, J. T., Petzold, L., Wright, M.: Applied mathematics at the U.S. Department of Energy: Past, present and a view to the future. Lawrence Livermore National Laboratory, (May 2008)

8. Buckwar, E., Horváth-Bokor, R., Winkler, R.: Asymptotic mean-square stability of two-step methods for stochastic ordinary differential equations, *BIT Numerical Mathematics*, 46(2) (2006), pp. 261–282.
9. Castell, F.: Asymptotic expansion of stochastic flows, *Probab. Theory Related Fields*, 96 (1993), pp. 225–239.
10. Castell, F., Gaines, J.: An efficient approximation method for stochastic differential equations by means of the exponential Lie series, *Math. Comput. Simulation*, 38 (1995), pp. 13–19.
11. Castell, F., Gaines, J.: The ordinary differential equation approach to asymptotically efficient schemes for solution of stochastic differential equations, *Ann. Inst. H. Poincaré Probab. Statist.*, 32(2) (1996), pp. 231–250.
12. Evans, L.C.: An introduction to stochastic differential equations, downloadable from the web, <http://math.berkeley.edu/~evans>
13. Gaines, J., Lyons, T.J.: Random generation of stochastic area integrals, *SIAM J. Appl. Math.*, 54(4) (1994), pp. 1132–1146.
14. Gaines, J. G., Lyons, T. J.: Variable step size control in the numerical solution of stochastic differential equations, *SIAM J. Appl. Math.*, 57(5) (1997), pp. 1455–1484.
15. Gillespie, D.T.: The chemical Langevin equation, *Journal of Chemical Physics*, 113(1) (2000), pp. 297–306.
16. Gilling, H., Shardlow, T.: SDELab: stochastic differential equations with MATLAB, MIMS EPrint: 2006.1, ISSN 1749-9097.
17. Glasserman, P.: Monte Carlo methods in financial engineering, *Applications of Mathematics, Stochastic Modelling and Applied Probability*, 53, Springer, 2004.
18. Heston, S.L.: A closed-form solution for options with stochastic volatility with applications to bond and currency options, *Review of Financial Studies*, 6(2) (1993), pp. 327–343.
19. Higham, D.J.: An algorithmic introduction to numerical simulation of stochastic differential equations, *SIAM Review* 43 (2001), pp. 525–546.
20. Karatzas, I., Shreve, S.E.: *Brownian motion and stochastic calculus*, Springer, 1991.
21. Karlin S., Taylor, H.M.: *A second course in stochastic processes*, Academic Press, 1981.
22. Kloeden, P.E., Platen, E.: *Numerical solution of stochastic differential equations*, Springer, 1999.
23. Knuth, D.E.: *The art of computer programming*, vol 2: Seminumerical algorithms, Addison-Wesley, Reading, Mass., Third edition, 1998.
24. Lévy, P.: Wiener’s random function and other Laplacian random functions, *Second Symposium of Berkeley. Probability and Statistics*. UC Press, 1951, pp. 171–186.
25. Lord, R., Koekoek, R., van Dijk, D: A comparison of biased simulation schemes for stochastic volatility models, *Tinbergen Institute Discussion Paper TI2006-046/4*, 2006.
26. Lord, G., Malham, S.J.A., Wiese, A.: Efficient strong integrators for linear stochastic systems. *SIAM J. Numer. Anal.* 46(6) (2008), pp. 2892–2919.
27. Lyons, T.: Differential equations driven by rough signals, *Rev. Mat. Iberoamericana*, 14(2) (1998), pp. 215–310.
28. Lyons, T., Victoir, N.: Cubature on Wiener space. *Proc. R. Soc. Lond. A* 460 (2004), pp. 169–198.
29. Magnus, W.: On the exponential solution of differential equations for a linear operator, *Comm. Pure Appl. Math.*, 7 (1954), pp. 649–673.
30. Marsaglia, G.: Improving the polar method for generating a pair of random variables, *Boeing Sci. Res. Lab.*, D1-82-0203, 1962.
31. Milstein, G. N.: *Numerical integration of stochastic differential equations, Mathematics and its applications*, Kluwer Academic Publishers, 1994.
32. Malham, S.J.A., Wiese, A.: Stochastic Lie group integrators. *SIAM J. Sci. Comput.* 30(2) (2008), pp. 597–617.
33. Moro, B.: The full Monte, *Risk Magazine* 8(2) (February, 1995), pp. 57–58.
34. Newton, N. J.: Asymptotically efficient Runge–Kutta methods for a class of Itô and Stratonovich equations, *SIAM J. Appl. Math.*, 51 (1991), pp. 542–567.
35. Oksendal, B.: *Stochastic differential equations: An introduction with applications*, Sixth edition, Springer, 2003.
36. Rydén, T., Wiktorsson, M.: On the simulation of iterated Itô integrals, *Stochastic processes and their applications*, 91 (2001), pp. 151–168.
37. Schurz, H.: A brief introduction to numerical analysis of (ordinary) stochastic differential equations without tears, in *Handbook of Stochastic Analysis and Applications*, V. Lakshmikantham and D. Kannan, eds., Marcel Dekker, 2002, pp. 237–359.

38. Strichartz, R. S.: The Campbell–Baker–Hausdorff–Dynkin formula and solutions of differential equations, *Journal of Functional Analysis*, 72 (1987), pp. 320–345.
39. Stump, D.M., Hill, J.M.: On an infinite integral arising in the numerical integration of stochastic differential equations, *Proc. R. Soc. A*, 461 (2005), pp. 397–413.
40. Sussmann, H.J.: Product expansions of exponential Lie series and the discretization of stochastic differential equations, in *Stochastic Differential Systems, Stochastic Control Theory, and Applications*, W. Fleming and J. Lions, eds., Springer IMA Series, Vol. 10, 1988, pp. 563–582.
41. Van Kampen, N.G.: *Stochastic processes in physics and chemistry*, North–Holland, Elsevier Science Publishers, 1981.
42. Wiktorsson, M.: Joint characteristic function and simultaneous simulation of iterated Itô integrals for multiple independent Brownian motions, *Ann. Appl. Probab.*, 11(2) (2001), pp. 470–487.